Results of 9/10/08 discussion with Javier, Matt, Peng, Mike and Tristan.

Current

We currently have two extremes:

1. TimetableProfile does fast/simple calculations. The regular flaw/violation detector will always report flaws with the resulting profiles, and the Timetable flaw/violation detector can be used to only report violations (ignore flaws)
2. FlowProfile (and IncrementalFlowProfile, which does the same thing, just a bit more efficiently) does the best possible calculations based on the avialable constraints (TODO: what does 'best possible' really mean).

Note that in general flaws occur when LB < 0 and violations when UB < 0 (for the level can't be below 0 case, at least).

Improvements

Possibilities, in a rough priority order:

1. (general case) Tighten up Timetable profile. Take care of the obvious places in where flaws needn't exist (precedence chains, for example).
2. Can we reduce the set of profiles computed from 4 down to 2 or 1 in various situations:
    1. (special case) If no variable production, we only need 2 of the profiles (upper upper, and lower lower)
    2. (special case) Can we reduce from 4 to 2 if we make standard assumptions about instantaneous or cumulative production/consumption.
    3. (special case) If Timetable profile is used with Timetable flaw/violation detection, can we just use a single profile (ie the one used for violations, not flaws)? Does this only work if we make assumptions about variable production/consumption and/or non-existence of instantaneous and/or cumulative limits.
    4. (special case) If we only have production, or only have consumption, do we need less information/computation?
3. (special case) Special case unary resources. There should be a way to make this as fast as timelines, I suspect.
4. (special case) Javier mentioned there may be additional insight in the Reusable case, where you know that every consumption will be followed by an equal production, and vice-versa.
5. (special case) Grounded plans:
    1. I originally thought grounding meant that we would either:
        1. Only report violations after everything is grounded (easy, but too late?). It appears that using Timetable profile AND flaw/violation detector gets similar results, but is better, because not everything needs to be grounded.
        2. To calculate flaws/violations, create a profile by stepping through time and greedily choosing a time for each transaction (subject to all constraints), and reporting whether or not there is a violation, or something...
    2. Javier's notion of grounding, as I understand it, means that your plan is grounded, but you use information from that for no-good detection to figure out how to handle violations that do appear. Edge-finding algorithms may help here.
6. (general case) Will edge-finding algorithms help in other cases?